JAVASCRIPT NOTE

Struttura:

La cosa sicuramente più semplice di uno script javascript è descriverne la struttura.
Gli script possono essere sia interni (scritti all'interno della pagina HTML) che esterni (in fogli .js), un pò come gli style o css.

Script interni:

Gli script interni alla pagina HTML possono essere inseriti sia fra i tag head che fra i tag body. La differenza sta nel fatto che se inseriti fra i tag head saranno attivati prima ancora che la pagina venga mostrata al navigatore per poi essere richiamati al momento opportuno (ad esempio con l'evento onClick). Al contrario fra i tag body lo script sarà attivato solo quando trovato dal browser.
Per inserirli internamente alla pagina dobbiamo scrivere le seguenti righe:

```
<script language="JavaScript" type="text/javascript">
<!--
Qui inseriamo il nostro script
-->
</script>
```

I tag <script> e </script> vogliono dire inizio e fine script.
I simboli <!-- e --> sono rispettivamente inizio e fine commento HTML. Sono fondamentali per il semplice motivo che alcuni browser (pochissimi) non leggono gli script, ed in questo caso vedrete a video il codice script come normale testo. Il fatto che inseriate il commento scongiurerà questa eventualità.

Script esterni:

L'utilizzo dei fogli esterni consente una velocità superiore di caricamento della pagina web, perché Il foglio .JS sarà scaricato e depositato nei file temporanei del browser del navigatore, per poi essere richiamato senza un'ulteriore caricamento ogni volta che vorremo utilizzare lo stesso script (su più pagine diverse).
La procedura per la creazione del foglio esterno javascript è simile ai fogli CSS, ossia creiamo un foglio e lo salviamo con estensione .js
La prima riga sarà il commento <!-- e l'ultima riga sarà il fine commento --> (scongiurerà di vedere a video il testo dello script nei browser che non supportano javascript).

Per richiamare la pagina dello script sarà sufficente inserire le seguenti righe all'interno della pagina HTML:

```
<script language="JavaScript" type="text/javascript" src="nomefoglio.js">
</script>
```

Quando il navigatore si collegherà al vostro sito scaricherà sia il foglio .JS che la pagina html. Lo script sarà letto dal browser nel momento in cui troverà il richiamo del foglio .JS

Posizione:

Una caratteristica fondamentale è il fatto che i browser leggono la pagina HTML dall'alto verso il basso ed eseguono il codice quando lo trovano. Una funzione può essere richiamata all'interno di una tabella od un

box div solo se dichiarata in precedenza.

Dove inserire gli script:

- Head
  Fra i tag head si inseriscono gli script che vogliamo far caricare prima che la pagina web venga visualizzata, rendendo accessibili le funzioni in tutto il foglio html.
- Body
  Se la funzione viene usata solo in un punto del foglio possiamo inserirla prima del suo uso nel foglio html. Lo script potrà essere utilizzato soltanto dopo che il browser lo ha trovato e fatto girare.
- Esterni
  I fogli esterni vengo utilizzati per snellire la pagina web e per rendere disponibile lo script a più pagine.

Richiamo:

Uno script javascript può essere eseguito per girare nel luogo in cui si trova (all'interno di una tabella o box), oppure può essere utilizzato per dare una risposta ad un'evento del navigatore. Per evento intendiamo un'azione del navigatore sopra un oggetto, ad esempio un clic su un'immagine, il passaggio del mouse su una scritta ecc.

body:

Un'attenzione particolare va mostrata all'evento di caricamento o uscita pagina. Spesso può essere utile attivare uno script appena caricata una pagina web (per mostrare qualcosa) oppure appena una pagina viene lasciata (magari per mostrare un pop-up).

Se una funzione deve attivarsi appena dopo il caricamento visivo della pagina web lo possiamo fare inserendo un'istruzione nel tag body come nell'esempio:

<body onLoad="nomefunzione()">

In questo caso prima di caricare il foglio verrà eseguito lo script.
Solito discorso quando chiudiamo il documento:

<body onUnLoad="nomefunzione()">

Variabili javascript:

Anche javascript utilizza le variabili per manipolare i dati dello script.
Per variabile intendiamo tutte le lettere o parole a cui associamo un valore.
Il valore può essere numerico, letterale o booleano (true o false), e potrà variare nel corso di una funzione.

Per utilizzare una variabile dobbiamo dichiararla prima di usarla in questo modo:
Var a
Da adesso il pc sa che può usare la variabile a.

Per associare un valore ad una variabile la dobbiamo inizializzare (solitamente lo si fa quando si dichiara) vediamo l'esempio per capire:
Var a = 5 se numerica.
Var a = "ciao" se letterale (stringa).
Var a = true se booleana.

Non è obbligatorio utilizzare 'var' se stiamo inizializzando una variabile, ma è bene metterla per semplificare la comprensione del ciclo da parte nostra e del browser.

Tipi di variabili:

Le variabili possono essere di tre tipi:
Number Numerica Es. Var x = 10
String Letterale Es. Var x = "ciao" (ricordarsi le virgolette)
Boolean True o False (vero o falso) Es. Var x = true

Funzioni javascript:

Alla base di ogni script troviamo le funzioni. Le funzioni non sono altro che un insieme di codice JavaScritp racchiuso fra parentesi graffe con un nome per farne richiamare l'utilizzo.
All'interno delle funzioni possiamo inserire tutte le operazioni necessarie allo script.
All'interno di uno script possiamo inserire più funzioni per poi farle interagire.

Esempio:
Function a(){
...codice JS....
}

Dichiarazione:

Per dichiarare una funzione dobbiamo utilizzare la parola chiave 'function' seguita dal nome della funzione e dai suoi parametri racchiusi in parentesi tonde e separati da una virgola.

Esempio:
Function nomefunzione(a,b,c){
Var d=a+b+c
return d
}
I parametri a, b, c (che possono assumere qualsiasi nome) verranno utilizzati all'interno della funzione, non sono obbligatori.
Per restituire il valore risultante dalla funzione dobbiamo inserire 'return' alla fine del codice seguito dal nome della variabile che restituiamo, ma non è obbligatorio.

Chiamata:

Per chiamare una funzione dobbiamo scrivere il suo nome seguito dalle parentesi tonde con i parametri al suo interno.

Esempio:
document.write("il numero risultante è "+ qualsiasinome(4,10,30))
In questo caso il risultato sarà:
il numero risultante è 44

Metodi:

I metodi sono funzioni già presenti in JavaScript, per esempio clic()
Per capire meglio vai nella sezione Oggetti, proprietà e metodi, oppure vedi la tabella Tabella oggetti, proprietà, metodi e gestori eventi.

Return:

Il comando return viene utilizzato per restituire un valore ben preciso risultante dalla funzione. Con questo non vogliamo dire che sia obbligatorio, tutt'altro.
Quando usiamo return la funzione si interrompe, e prosegue con le righe di codice successive. Questo è uno dei motivi per cui possiamo usare return, ossia ci permette di uscire da una funzione senza interrompere il programma JavaScript.
La sintassi è le seguente:
return nomevariabile;
oppure
return (nomevariabile);
Dove nomevariabile è il nome della variabile che vogliamo restituire.

CICLI JAVASCRIPT

Come in ogni linguaggio di programmazione anche JavaScript possiede i cicli, ossia delle istruzioni che ciclicamente possono modificare o interagire con tutti gli oggetti dello script. Vediamo come si chiamano e cosa possono fare.

IF

Vedi anche l'operatore di confronto
Il ciclo IF (in italiano SE) pone una scelta, ossia 'se vero' esegui in un modo oppure 'se falso' esegui in un altro modo. I cicli IF si possono annidare fra loro, ossia un ciclo IF ne può contenere un altro.

Esempio:

```
if (a==b){
document.write("uguali")
}
else{
document.write("diversi")
}
```

Le parentesi graffe si usano per raggruppare più informazioni, quindi se ci sono più azioni da fare in una delle ipotesi mettiamo le graffe, se c'è solo un'azione le possiamo anche omettere.
La clausola Else { } è facoltativa, va usata nel caso ci sia un'alternativa al ciclo IF.

Questo è l'esempio pratico, anche se con l'aiuto di un form, vediamo il codice:

```
<script>
<!--
function confronto(form){ //dichiariamo la funzione confronto
if (form.a.value && form.b.value) //chiediamo se A e B sono inseriti
{ //racchiude tutte le azioni attivate se A e B sono inseriti
if (form.a.value==form.b.value) //sono inseriti e ora chiediamo se A e B sono uguali
form.c.value="uguali" //sono uguali e lo scriviamo
else //se non sono uguali....
form.c.value="diversi" //sono diversi e lo scriviamo
} //Conclude tutte le azioni
else //se non sono inseriti....
alert("devi inserire i valori") //manda l'alert con il messaggio
}
-->
</script>
```

Ed adesso il codice del form:

```
<form name="d" action="#">
<b>A</b> <input size="10" name="a"><br>
<b>B</b> <input size="10" name="b"><br>
<textarea name="c" rows="1" cols="10"></textarea><br>
<input onclick=confronto(this.form); type="button" value="Aziona la funzione confronto"><br>
<input type="reset" value="Reset" name="cancella">
</form>
```

Uno strano esempio:

```
if (!ciao) {
ciao = 5
}
```

In questo strano esempio chiediamo se la variabile 'ciao' esiste, se non esiste la creiamo e la inizializziamo a 5.

FOR

Il ciclo for esegue un ciclo di informazioni fino a quando la condizione iniziale non diviene falsa.
La sintassi è la seguente:

```
for (espressione iniziale; condizione; aggiornamento){
....operazioni....
}
```

Facciamo un esempio pratico per rendere meglio l'idea:

```
<script>
for (a=0; a<=10; a++){
document.writeln("ciao "+a)
}
</script>
```

Ottiene come risultato:

ciao 0 ciao 1 ciao 2 ciao 3 ciao 4 ciao 5 ciao 6 ciao 7 ciao 8 ciao 9 ciao 10

FOR IN

Il ciclo for in serve per vedere tutte le proprietà di un oggetto.
La sua sintassi è la seguente:

```
for (var in nomeoggetto) {
...operazioni...
}
```

Anche qui sarà utile un esempio per capire come funziona:

Questo è il codice:

```
<script>
function abc(oggetto, nomeOggetto){
var risultato = ""
var contatore=1
for (var ciascunElemento in oggetto) {
if (contatore <= 4) {
risultato = risultato + (nomeOggetto
+ "." + ciascunElemento + " = "
+ oggetto[ciascunElemento]
```

```
+ "n")
}
contatore++
}
alert("Ecco il contenuto dell'oggetto "
+ nomeOggetto
+ ": n"
+ risultato)
}

</script>
<form name="mioForm">
<input type="button" value="ispeziona l'oggetto document utilizzando for...in" onClick='abc(document,
"document")'>
</form>
```

Il ciclo dell'esempio osserva alcune caratteristiche della nostra pagina.

WHILE

Il ciclo while dura fino a quando la condizione è vera. Per far questo dobbiamo necessariamente far variare la condizione all'interno del ciclo.
La sua sintassi è la seguente:

```
while(variabile condizione valore){
..operazioni..
modificavariabile}
```

Questo è il codice:

```
<script>
<!--
function confronto(){
var b = h.a.value
if (b<=5 && b!=''){
while(b<=5){
h.c.value=h.c.value+("num"+b)
b++}
}
else
alert("inserire un valore numerico non superiore a 5")
}
-->
</script>

<form name="h" action="#">
<b>A</b><input size="10" name="a">Inserisci un valore da 0 a 5
```

<input onclick=confronto(); type="button" value="Aziona">
<input type="reset" value="Reset" name="cancella">
</form>

In questo caso abbiamo inserito un ciclo while all'interno di un ciclo if, utilizzando i dati che abbiamo inserito in un form.

Nell'esempio il ciclo while è costituito dalle seguenti righe:
while(b<=5){
h.c.value=h.c.value+("num"+b)
b++}
In sostanza chiediamo se la variabile b è minore o uguale a 5, eseguiamo l'operazione ed infine aumentiamo b di 1. Fino a quando b sarà sempre minore od uguale a 5 eseguiremo l'operazione.

DO WHILE

Il ciclo do while è molto simile al ciclo while. La differenza sostanziale è che il ciclo while può non essere eseguito, questo accade se la condizione risulta falsa fin dall'inizio. Il ciclo do while si esegue sempre, almeno per una volta.
Questo perché il ciclo do while inserisce prima le azioni da fare e dopo la condizione. Il browser esegue le prime istruzioni, poi legge la condizione e se è sempre vera riesegue le istruzioni.
Vediamo il codice:
do {
...azioni...
}
while(condizione)

SWITCH

L'istruzione switch non è un vero e proprio ciclo. Switch viene usato quando abbiamo più alternative da vagliare e non vogliamo inserire più cicli if annidati fra loro.
Supponiamo per esempio di inserire una variabile e di dover agire in maniera diversa se questa variabile corrisponde a due valori. Con l'istruzione if dovremo scrivere due cicli annidati, con switch ne basta uno.
Questo è il codice:
switch (variabile) {
case "x":
azioni
break
case "xx":
azioni
break
}
Vediamo un esempio pratico:

Questo è il codice:

```
<script>
<!--
function inserire(){
var b = g.a.value
switch (b){
case"a":
alert("hai inserito a")
break
case"b":
alert("hai inserito b")
break
case"c":
alert("hai inserito c")
break
default:
alert("non hai inserito ne a, ne b, ne c")
break
}
}
-->
</script>

<form name="g" action="#">
<b>A</b> <INPUT size="10" name="a"> <input onclick=inserire(); type="button" value="Aziona"> </form>
```

DEFAULT racchiude tutte le alternative che non sono state vagliate dal ciclo.

Oggetti javascript:

JavaScript è un linguaggio basato sugli oggetti. Definiamo oggetto:
Per oggetto intendiamo qualunque cosa serva per creare la nostra pagina web:
- oggetti definiti mediante HTML (es. img...)
- oggetti definiti dal browser (es. navigator...)
- oggetti di JavaScript (es. date...)
- oggetti creati dall'utente (con new)

Accedere agli oggetti

Tutti gli oggetti di JavaScript partono da Window fino ad arrivare alla proprietà, al metodo o al gestore evento dell'oggetto. Per accedere ad un oggetto dobbiamo descrivere il percorso per arrivarci. Per far questo iniziamo dal punto più alto conosciuto (solitamente window o document) fino a scendere nei dettagli. Il tutto con dei punti che separino le parole (Un esempio pratico esterno è il caricamento del sito: www.allwebfree.it)
Esempio:
Document.nomeform.valuepulsante In questo caso accedo ad un form.

se in pratica ho un form di questo genere:
<form name="nomeform">
<input size="10" name="a">
.... Per modificare il valore del campo input di nome "a" all'interno dello script devo inserire un percorso del tipo nomeform.a.value=""

(N.B. Esiste un altro metodo per arrivare ai campi dei form, puoi vederlo qui)

Proprietà:

Le proprietà sono tutti gli attributi che distinguono gli oggetti. Per fare un esempio basta pensare ad una immagine, che ha come proprietà src, name, border ecc.

Metodi:

Il metodo è una funzione particolare già presente in JavaScript che descrive il comportamento di un oggetto. Per capire immaginate l'oggetto pulsante con il metodo click() che non è il gestore di eventi onClick().

La differenza fra i metodi ed i gestori di eventi, è che il metodo descrive un'azione da applicare ad un oggetto quando lo script entra in azione, mentre il gestore di eventi risponde ad un'azione di chi naviga.

Per descrivere un metodo dobbiamo in precedenza dire il nome dell'oggetto a cui viene associato, oppure inserirlo direttamente nella riga HTML che descrive l'oggetto.
Esempio:
back

Codice:

back

In questo caso il metodo back() è preceduto dall'oggetto window.history